Submitted By : Afjal J. Ansari
Internship ID : APSPL2517969
Department : Cybersecurity
Organization : ApexPlanet software Pvt.Ltd
Co-ordinator : Kundan Kumar
Date of Submission : September 18, 2025

1. CIA – Not the Spy Guys
CIA in cybersecurity = **Confidentiality, Integrity, Availability**
Acts like a **checklist** for any security project
Every system should satisfy all 3

2. Confidentiality
Definition: Only authorized users can access info/resources
How to enforce:
- Authentication & Authorization
- Multi-Factor Authentication (MFA)
- Encryption

Example: A student logs into a portal → gets marks only if authorized.
Unauthorized users → blocked.
Privacy vs Confidentiality:
- Privacy (modern view) = user rights (consent, right to be forgotten).
- Confidentiality = access control, secrecy.

3. Integrity
Definition: Data must remain accurate, trustworthy, and unchanged.
Example: Order of 100 widgets must not become 100,000.
Threats:
- Tampering with records/logs
- Modifying or deleting data

Countermeasures:
- Hashing & checksums
- Digital signatures
- Monitoring & alerts for tampering

Think: "System must stay true to itself."

4. Availability
Definition: Authorized users must access resources when needed.
Example: Student should access exam portal during exam time.
Threats:
- Denial of Service (DoS) attacks
- Distributed DoS (DDoS) – flooding server with fake requests
- System crashes or failures

Countermeasures:
- Load balancing
- Backups & redundancy
- Firewalls & anti-DDoS tools

5. Final Checklist
Whenever making a security plan/project, ask:
✅ Is my system **Confidential** (only right people see data)?
✅ Is my system maintaining **Integrity** (data isn't altered)?
✅ Is my system **Available** (up & running when needed)?
If **yes to all three → CIA secured** 🔐

Types of Threats

1. Clickjacking
Trick user into clicking on hidden/invisible elements.
Example: Fake "Play" button → secretly turns on webcam.
Used for spying, blackmail, or data theft.

2. Phishing
Hacker pretends to be trusted (bank, company, etc.).
Fake emails, websites, SMS.
Goal: steal credentials or sensitive info.
Example: "Fix your account" → takes you to fake login page.

3. Identity Theft
Hacker uses stolen personal info (name, DOB, address).
Opens credit cards, loans, or impersonates victim.
Victim faces debts & legal trouble.

4. Credential Stuffing
Hackers reuse stolen usernames/passwords on multiple sites.
Works because people often reuse passwords.
Prevention: unique password per account + MFA.

5. DDoS (Distributed Denial of Service)
Flood server with fake traffic → crashes/slowdown.
Uses botnets (infected devices worldwide).
Example: gov. site normally handles 1000 visits → hit with millions.

6. Brute Force
Guessing passwords by trying all combinations.
Automated scripts test common passwords.
Defense: complex passwords, lockout systems, MFA.

7. Eavesdropping
Secretly listening to unencrypted communication.
Common in public Wi-Fi without encryption.
Hacker can read sensitive data (emails, passwords).

8. MITM (Man-in-the-Middle)
Like eavesdropping, but hacker can **alter** data.
Example: Seller sends bank details → hacker replaces with his own.
Victim unknowingly pays hacker.

9. Typosquatting
Fake sites with misspelled domains (gooogle.com).
User lands on malicious site → malware/cryptojacking installed.
Doesn't need password theft to work.

10. Insider Threat
Attack from within the organization.
Example: Angry ex-employee plants a logic bomb.
Harder to detect because access is legitimate.

11. Social Engineering
Attacking humans, not machines.
Example: Dropping USB labeled "Salary Bonus 2025."
Curiosity leads employees to plug it in → malware spreads.

12. SQL Injection
Exploiting weakly coded websites.
Malicious SQL commands trick site into giving data or admin access.
Can steal or wipe databases.

13. DNS Poisoning
Hackers tamper with DNS records.
Even if user types correct site (paypal.com), they land on a fake one.
Credentials get stolen.

14. Drive-by Download
Malware downloads automatically from malicious ads/pop-ups.
Doesn't always need user click.
Often exploits outdated browsers.

15. XSS (Cross-Site Scripting)
Hacker injects malicious code into trusted websites.
Runs in victim's browser when page loads.
Example: Blog comment section runs hidden script.

16. IoT Exploitation
Hijacking smart devices (TVs, cameras, speakers).
Used for spying or surveillance.
Often exploited in authoritarian regimes.

17. Zero-Day Exploit
Exploits unknown software vulnerabilities.
No fix available until developers release a patch.
Famous case: Stuxnet (2010) attacked Iran's nuclear facilities.

18. Supply Chain Attack
Hackers target software updates instead of individual users.
Inject malware into official updates.
Victims (companies, governments) unknowingly install compromised updates.
Example: SolarWinds attack.

Social Engineering – Hacking Humans
1. Why Target Humans?
Humans = weakest link in security.
Easier to trick a person than hack a machine.
Social engineering = exploiting psychology (not technology).
Two main levers hackers use:

- **Greed** (wanting reward, upgrade, free stuff).
- **Fear** (scared of loss, urgency, pressure).

2. Scenario 1 – Credential Theft (Greed-based)
Attacker does research (social media, LinkedIn, Google).
Sends a **spear phishing email** (very targeted).
Example: Fake "new laptop upgrade" link → typo-squatted site.
Victim logs in → ID & password stolen.
Prevention:
Secure DNS (Quad9) → blocks known malicious sites.
User awareness → verify links, check spelling.
Critical thinking → "Too good to be true? Probably is."

3. Scenario 2 – RAT Installation (Fear-based)
Attacker phones victim, pretends to be "tech support."
Claims malware detected → urgent threat ("your files will be deleted!").
Victim panics → downloads fake "disinfection tool."
Actually installs a **RAT (Remote Access Trojan)** → attacker gets full control.
Prevention:
Education → real software companies don't cold-call.
Secure DNS → block malicious downloads.
Critical thinking → hang up if pressured.
MFA → adds extra protection against account misuse.

4. Scenario 3 – Deepfake Attack (Fear + Urgency)
Goal: Steal corporate intellectual property.
Attacker researches victim → finds videos, uses **AI deepfake voice generator.**
Calls victim's assistant → leaves urgent voicemail ("send files to personal email NOW or we'll lose our jobs").
Victim trusts voice, urgency → sends confidential data to hacker.
Real case: Bank lost **$35M** due to voice deepfake scam.
Prevention:
User training → don't trust voice blindly.
Verify requests through another channel (call back official number).
Strict rules → never send company data to personal accounts.

5. Key Takeaways
Social engineering = **attack on psychology, not technology.**
Solutions:
- **Tech tools:** Secure DNS, MFA.
- **Human tools:** Awareness, critical thinking, training.

No firewall exists for the human brain → must build awareness.
Always stop & think: **"Does this make sense, or am I being rushed/manipulated?"**

Wireless Attacks – Deauthentication & Jamming
1. Deauthentication Attack (Wi-Fi DoS)
What happens? You get randomly disconnected from Wi-Fi again and again.
How? Hackers abuse **management frames** (connect/disconnect signals).
Old **802.11 Wi-Fi standard** sent these frames **unencrypted** → anyone nearby can fake them.
Tools: **airodump-ng** (scan Wi-Fi devices) + **aireplay-ng** (send fake deauth frames).
Victim's device keeps getting kicked off → **Denial of Service (DoS).**
Fixes:
Newer Wi-Fi standards (**802.11ac & later**) encrypt management frames.
But some frames (beacons, probes, association) must stay clear for devices to connect.

2. Wireless Jamming Attack (Noise Overload)
What happens? Wi-Fi becomes useless → no connection possible.
How? Attacker floods the radio frequency (2.4 GHz / 5 GHz) with junk signals → lowers **signal-to-noise ratio**.
Can be:

- **Constant jamming** (continuous noise).
- **Random jamming** (at unpredictable times).
- **Reactive jamming** (starts only when legit traffic appears).

Result: Wi-Fi devices can't "hear" the access point.
Fun fact: Microwaves and fluorescent lights sometimes unintentionally jam Wi-Fi too!

3. Detecting the Attacker (Fox Hunt 🦊)
Jammers must be **physically close** to the Wi-Fi they're attacking.
To find them:

- Use **directional antennas** → point toward the signal.
- Use an **attenuator** to reduce signal strength (helps pinpoint direction).

Security pros call this a **"fox hunt"** – tracking down the jammer like hunting a sneaky animal.

4. Big Takeaway 📝
Both attacks = **Denial of Service (DoS).**
Deauth = exploiting **Wi-Fi protocol weakness.**
Jamming = exploiting **physics (radio waves).**
Defense: Upgrade to latest Wi-Fi standards, monitor networks, and physically hunt signal sources.

👉 In short: **Hackers can either trick your Wi-Fi into kicking you out (deauth) or scream so loud on the frequency (jamming) that your Wi-Fi can't talk at all.**

Insider Threats – Danger From Within
1. Why It's Tricky 🚨
Insiders = already trusted people.
They know the **systems, servers, and processes.**
Don't always need admin rights → even normal employees have useful knowledge.
They can act **slowly over time**, gathering intel unnoticed.
Harder to detect than outsiders because they "belong."

2. Types of Insider Threats 🦹‍♂️
Malicious insiders → steal, sabotage, or sell data.
Recruited insiders → bribed by attackers (sometimes in cryptocurrency 💰).
Careless insiders → not evil, just make mistakes (leaving USB, weak passwords).
Rogue employees → angry after being fired, use old credentials for revenge.

3. Real-World Example 💡
Ransomware gangs sometimes **offer employees money** to plant malware.
One infection = **millions in profit** for attackers.
For a greedy insider, it looks like a "business deal."

4. Defense Strategy 🛡️
Security fundamentals first:
- Strong access controls (least privilege).
- Monitoring + logging (catch unusual behavior).
- Regular security awareness training.

Always keep backups → if files are stolen or encrypted, you can recover.
Zero trust mindset: Never fully trust, always verify.

5. Big Takeaway 📝
Insiders can be more dangerous than outsiders.
Prevention is tough → focus on **limiting damage** and **detecting early.**
Remember: A hacker can break your walls, but an insider already has the keys.

👉 In short: **Protecting against insiders = protecting against "trusted enemies." Build strong defenses, watch carefully, and always back up.**

Installing VirtualBox on Windows 11
1. Download VirtualBox 📥
Open your browser → search **"VirtualBox"**.
Click the official site: **virtualbox.org**.
Select **Download**.
Choose **Windows Host** (for Windows 11).
.exe installer file will download.

2. Run the Installer ⚙️
Right-click the downloaded .exe → **Open**.
Setup wizard opens → click **Next**.
Accept the license terms.
Leave installation location as default → **Next**.
Allow creation of Start Menu + Desktop shortcut → **Next**.
Click **Install**.

3. Complete Installation ✅
After install, click **Finish**.
A desktop shortcut appears.
Or search "**Oracle VirtualBox**" in Start Menu.
Open it → you're ready to use VirtualBox!

4. Quick Tip 💡
VirtualBox helps you run **virtual machines** (Linux, Kali, Ubuntu, etc.) on your Windows PC.
Useful for **cybersecurity labs** and **safe testing** without harming your main system.

Installing Kali Linux on VirtualBox (Windows 11)
1. Prerequisites 🖥️
Enable Virtualization:
Open **Task Manager → Performance → CPU → Virtualization**.
If *Disabled*: Enter **BIOS** (F2/F10/DEL/Esc at startup). Enable **Intel VT-x / AMD-V**.
Disable Hyper-V (can conflict with VirtualBox):
Open **Command Prompt (Admin)** → run:
bcdedit /set hypervisorlaunchtype off
Restart PC.

2. Install VirtualBox & Extension Pack 📦
Go to **virtualbox.org → Downloads**.
Select **Windows Host** → download installer.
Install with default settings → Finish.
Download & install **Extension Pack** (Tools → Extensions → Add).
Enables: USB 3.0, webcam, clipboard, drag-and-drop.

3. Download Kali Linux ISO 🐉
Go to **kali.org/get-kali**.
Choose **Installer Image (ISO)** → Download **64-bit ISO**.
(Tip: Installer image = full control; prebuilt VM = quick but less flexible).

4. Create Kali VM in VirtualBox ⚙️
Click **New** → Name: *Kali Linux*.
Type: **Linux** | Version: **Debian (64-bit)**.
RAM: Minimum 2GB → better 4GB+ (don't exceed half system RAM).
CPU: 2+ cores recommended.
Storage: 25GB (minimum), 50GB (better).
Settings Tweaks:
General → Advanced → Enable bidirectional **Clipboard & Drag-and-Drop**.
Display → Video Memory → **128MB**.
Storage → Attach Kali ISO under **IDE Controller**.
Network → **NAT** (default) or **Bridged** (for real IP on LAN).

5. Install Kali Linux 🛠️
Start VM → Select **Graphical Install**.
Steps:

- Choose Language, Country, Keyboard.
- Enter Hostname (any name).
- Create User + Password.
- Disk Partitioning → **Guided, Use Entire Disk → All files in one partition**.
- Install Base System (10–15 min).
- Choose Desktop Environment:
  - **XFCE (default, lightweight)**.
  - **GNOME (modern, needs 4GB+ RAM)**.
  - **KDE (Windows-like, resource-heavy)**.
- Install **GRUB bootloader** → Yes → Select Virtual Disk.
- Finish install → Reboot.

6. First Boot & Updates 🔄
Login with user credentials.
Full Screen: Press **Right Ctrl + F**.
Update system:
sudo apt update && sudo apt upgrade -y
Check version:
cat /etc/os-release
System monitor:
sudo apt install htop -y
htop

7. Optimize with Guest Additions 🛠️
Install (already in Kali repo):
sudo apt install -y virtualbox-guest-x11
reboot
Features: Better display scaling, clipboard sync, drag-and-drop, shared folders.

8. File Sharing Between Host & Kali 📁
Create **Kali_Share** folder on Windows.
VirtualBox → VM Settings → Shared Folders → Add new → Select folder.
Options: **Automount + Permanent**.
Reboot → Folder appears in Kali's file manager.

✅ **Final Result**
Kali Linux is installed & fully optimized in VirtualBox.
Supports **updates, drag-drop, shared folders, full-screen, and hacking tools**.

Installing & Linking Metasploitable 2 with Kali Linux (VirtualBox)

1. What is Metasploitable 2? 🐞

A **deliberately vulnerable VM**.

Used for **ethical hacking, penetration testing, and training**.

Think of it as your **personal hacking playground**.

2. Prerequisites 📦

Oracle VirtualBox installed.

Kali Linux VM already set up (for attacks).

Download **Metasploitable 2** VM.

3. Download & Extract Metasploitable 2 📥

Search **"Metasploitable 2 SourceForge"**.

Download from **sourceforge.net**.

File comes as a compressed archive → **Extract All**.

Inside, you'll find the **Metasploitable 2 virtual disk (.vmdk)**.

4. Import Metasploitable 2 into VirtualBox ⚙️

Open VirtualBox → Click **New**.

Name: *Metasploitable 2*.

Type: **Linux**, Version: **Other Linux (64-bit)**.

RAM: **512 MB** (default is fine).

CPU: **1 core**.

Choose **Use existing virtual hard disk file** → Browse & select the extracted .vmdk.

Finish setup.

5. Create a Virtual Hacking Lab 🌐

In VirtualBox → **File** → **Tools** → **Network Manager**.

Create a new **NAT Network** (e.g., *Hacking_Lab*).

Example: IP prefix → 192.168.1.0/24.

Enable **DHCP** → Apply.

6. Connect Kali & Metasploitable to Same Network 🔗

For **Kali VM**:

Settings → Network → Attach to: **NAT Network** → Choose *Hacking_Lab*.

For **Metasploitable 2 VM**:

Settings → Network → Attach to: **NAT Network** → Choose *Hacking_Lab*.

7. Start Metasploitable 2 🖥️
Boot VM.
Default login:
Username: msfadmin
Password: msfadmin
Run ifconfig → Note down IP (e.g., 192.168.1.4).

8. Verify Connection from Kali 🐉
Start Kali Linux VM.
Open terminal → Ping Metasploitable's IP:
ping 192.168.1.4
If you get replies → ✅ Kali can reach Metasploitable.

✅ **Final Setup**
Kali Linux = **Attacker machine**.
Metasploitable 2 = **Target machine**.
Both on same **virtual lab network**.
You can now practice **realistic penetration testing attacks** safely.

Linux File System Basics & Navigation
1. Directories vs Files 📂
Directory = same as a folder (just command-line terminology).
File = your data → docs, configs, pics, etc.
In Linux: *Everything is a file* (even devices & directories).

2. Root of the File System 🌳
Windows: C:\ = starting point.
Linux: / (forward slash) = root of all files.
Windows: separate drives (C:, D:).
Linux: all devices & drives live **under one root (/)**.

3. Useful Commands 💻
pwd → Print Working Directory (where am I?).
ls → List files in current directory.
ls -l → detailed list.
ls -lh → human-readable sizes (e.g., KB, MB).
ls -a → show hidden files.
Combine → ls -lah.
Shortcut in RedHat/Fedora/CentOS: ll = ls -l.

4. Moving Around 🌍
cd / → go to root directory.
cd etc → enter /etc folder.
cd ~ → go to home folder.
cd .. → go one step back.

5. Linux vs Windows 🆚
Windows → Few default folders (Windows, Program Files).
Linux → Many system-critical directories under /.
⚠️ Be careful → deleting system directories can break Linux.

✅ **Key Takeaways**
Directories = folders, Files = data.
Linux root / = Windows C:\.
Use pwd to check location.
Use ls + options to view details.
Use cd to move around directories.

Linux File Permissions Notes
1. Permission Groups 👥
Every file/directory has **3 groups of permissions**:
Owner → The user who owns the file.
Group → Users in the assigned group.
Others → Everyone else on the system.

2. Permission Types 🔑
Each group gets **3 types of permissions**:
r = read (4) → view contents.
w = write (2) → modify contents.
x = execute (1) → run file / list directory.
Example in ls -l:
drwxrwxr-x
d = directory (- means file).
First rwx = owner permissions.
Second rwx = group permissions.
Last r-x = others' permissions.

3. File vs Directory Example 📂
File → -rw-rw-r--
Owner: read + write.
Group: read + write.
Others: read only.
Directory → drwxrwxr-x
Owner: full control.
Group: full control.
Others: read + execute (can list files, but not modify).

4. Changing Ownership & Groups 👑
chown user file → Change owner.
chown user:group file → Change owner & group.
Example:
sudo chown travis myconfig.yaml
sudo chown travis:travismedia myconfig.yaml

5. Changing Permissions (chmod) 🔧
Uses **numbers** (based on 4+2+1 system):
r = 4, w = 2, x = 1
Add them up for each group.
Example:
rw- = 6
rwx = 7
r-- = 4
So → -rw-rw-r-- = **664**
Change with:
chmod 774 file.txt
chmod 755 foldername

6. Common Permission Settings 📋
Files → 644 (owner = rw, group = r, others = r).
Directories → 755 (owner = rwx, group/others = r-x).
Private Keys (AWS, SSH) → 400 (only owner can read).

✅ **Key Takeaways**
3 groups: owner, group, others.
3 permissions: read (4), write (2), execute (1).
Use chown for ownership, chmod for permissions.
Numbers (644, 755, 400) are shortcuts to define permissions.

Core Idea
Both dpkg and apt are package managers in Debian/Ubuntu-based Linux.
dpkg = the low-level tool (raw power, but manual).
apt = the higher-level tool (smart assistant, automatic).
Think of it like:
👉 dpkg is a screwdriver 🔧 (manual work).
👉 apt is a power drill ⚡ (does the same job but faster, smarter, easier).

📝 **Key Points from the Video**
1. What is dpkg?
The **fundamental package tool** for .deb files.
Handles install, remove, query of packages.
BUT: No dependency resolution! You install one package, and if it needs 5 more, you must install them manually.

2. What is apt?
Stands for **Advanced Package Tool**.
Works as a **front-end to dpkg**.
Can:
- Download packages from remote repositories 🌐
- Handle dependencies automatically ⚙️
- Provide progress bars, friendly output, etc.

3. Differences 🥊
Source of Packages:
- dpkg: Works with local .deb files only.
- apt: Fetches from repos or installs local .deb.

Dependencies:
- dpkg: No auto-fix.
- apt: Resolves automatically.

Usage:
- dpkg: More for developers/admins.
- apt: More for everyday users.

4. Syntax Examples 💻
Install with dpkg:
sudo dpkg -i package.deb
(If dependencies break, you need sudo apt install -f to fix.)
Remove with dpkg:
sudo dpkg -r package_name
Purge with dpkg:
sudo dpkg -P package_name

Install with apt:
sudo apt install package_name
Remove with apt:
sudo apt remove package_name

Install with apt:
sudo apt install package_name
Remove with apt:
sudo apt remove package_name
Purge with apt:
sudo apt purge package_name

5. Which One to Use? 🤔
For **end users** → Use apt (faster, easier, dependency-friendly).
For **special/manual cases** → Use dpkg (when dealing with raw .deb files).

🪄 **Rancho-Style Tip**
Imagine installing a car engine 🚙:
With dpkg, you get a box of engine parts. You assemble everything yourself.
With apt, a mechanic shows up with tools, extra parts, and even fixes broken screws.
So unless you love suffering (or learning Linux internals), stick to apt most of the time.

Core Idea
Networking commands help us **see how our computer connects** to the network and the internet.
Think of your PC like a person in a big city 🌆—these commands tell us:
What's my **address**? (ipconfig)
Who's my **neighbor**? (subnet mask)
Where's my **exit gate**? (default gateway/router)
How do I **find other people**? (nslookup → DNS)
Can I **reach them**? (ping)
Which **path do I take**? (tracert)

📝 **Key Commands from the Video**
1. ipconfig
Shows your **IP address** (like your house number).
Also shows **Subnet Mask** (who are in your colony/local network).
Shows **Default Gateway** (the router → first door out to the world).
Example:
ipconfig

2. ipconfig /all
More detailed version.
Shows **MAC Address** (like your computer's fingerprint).
MAC is used at **Layer 2 (Data Link Layer)**.
IP is used at **Layer 3 (Network Layer)**.
Example:
ipconfig /all

3. nslookup
Finds the **IP address of a domain name**.
Because we type google.com, but computers need 142.250.72.14.
DNS = Internet's phonebook.
Example:
nslookup nesoacademy.org
👉 Returns: 192.169.217.12

4. ping
Tests if a computer/server is **reachable**.
Sends 4 small packets and waits for reply.
If replies = ✅ reachable.
If no reply = ❌ unreachable.
Example:
ping www.facebook.com

5. tracert (trace route)
Shows the **path your packets travel** through routers (called hops).
Useful to check **where the connection is breaking**.
In Windows → tracert
In Linux → traceroute
Example:
tracert nesoacademy.org
👉 Shows 14 hops from your PC to their server.

🚀 **Rancho-Style Analogy**
Imagine sending a **letter to your friend abroad** ✉️:
ipconfig = Checking your own house address 🏠.
ipconfig /all = Adding fingerprint + postal ID too.
nslookup = Looking up friend's address in the phonebook 📖.
ping = Sending a "Hi, are you there?" text 📱.
tracert = Checking all airports & checkpoints your letter passes ✈️.

🏗️ **The OSI Model & Encapsulation (Simplified)**
👉 OSI Model is a **reference model**. It's like a recipe book 📖 telling us how to divide responsibilities in communication.
When data is sent, it **goes down the OSI layers** (encapsulation). At the receiver's end, it **goes up the layers** (decapsulation).

📦 **Encapsulation Steps (Sender → Receiver)**
Application, Presentation, Session (Layers 7–5)
Prepare the data.
Examples: Browsers, Email, Skype.
Output = Data.
🔑 Like **writing a letter** ✉️.
Transport Layer (Layer 4)
Breaks data into segments.
Adds source and destination ports.
TCP = reliable (registered parcel 📬).
UDP = fast but unreliable (postcard 📮).
🔑 Like writing the **apartment number** on the letter.
Network Layer (Layer 3)
Converts segment into packet.
Adds IP addresses (source & destination).
Protocol: IP.
🔑 Like adding the **house address** 🏠.
Data Link Layer (Layer 2)
Converts packet into frame.
Adds MAC addresses (device IDs).
Examples: Ethernet, Wi-Fi.
🔑 Like writing the **building gate number** 🏢.
Physical Layer (Layer 1)
Converts frame into bits (0s & 1s).
Sends via wires, light, or radio waves.
🔑 Like the **postman physically delivering it** 🚚.

🔄 **Decapsulation (Receiver Side)**
The receiver reverses the process:
Physical → Data Link → Network → Transport → Application.

🆚 **OSI vs TCP/IP Model**
OSI has 7 layers: Application, Presentation, Session, Transport, Network, Data Link, Physical.
TCP/IP has 4 layers: Application, Transport, Internet, Network Access.
TCP/IP is **practical** (used on the Internet).
OSI is **theoretical** (used for learning and design).

🧠 **Rancho Analogy**
Writing the message → Data.
Breaking into sentences → Segments.
Adding home addresses → Packets.
Adding building buzzer → Frames.
Sending via postman → Bits/signals.
💡 Whether it's email, video call, or WhatsApp meme → it's all **0s and 1s wrapped & unwrapped**!

🌐 TCP/IP Protocol Suite vs OSI Model

📌 OSI Model (Quick Recap)

7 Layers: **Physical, Data Link, Network, Transport, Session, Presentation, Application**.
Purely **theoretical** model (used for understanding).
Designed by **ISO (International Standards Organization)**.
Helps in learning **"what happens at each stage"** of communication.

📌 TCP/IP Protocol Suite

Known as **Internet Protocol Suite**.
Developed by **ARPANET**, funded by **DARPA (US Defence Agency)**.
Practical & implementable model (used in real internet).
Two common versions:

- **4-Layer Architecture**:
  - **Network Access Layer** (Physical + Data Link)
  - **Internet Layer** (IP, ICMP, IGMP)
  - **Transport Layer** (TCP, UDP, SCTP)
  - **Application Layer** (Application + Presentation + Session in one)
- **5-Layer Architecture**: Same as 4-layer, but **Physical & Data Link are separate**.

📌 Why the Difference?

OSI → **theory/documentation**.
TCP/IP → **practical implementation**.
Books vary (some show 4-layer, some 5-layer). **4-layer is most common in exams & industry.**

📌 Functions of TCP/IP Layers

Application Layer → Process-to-process delivery, user protocols like HTTP, FTP, SMTP, Telnet.
Transport Layer → Host-to-host delivery. TCP (reliable), UDP (fast).
Internet Layer → Source-to-destination delivery using **IP addressing** (IPv4, IPv6).
Network Access Layer → Node-to-node delivery in local network (MAC addressing, error & flow control).

📌 Data Encapsulation

Application → **Data**
Transport → **Segment**
Internet → **Packet**
Network Access → **Frame**
Physical (inside it) → **Bits**

📌 Extra Features

Supports **Client-Server** (centralized server + clients).
Supports **Peer-to-Peer** (no central server, all devices equal).
Used everywhere in the Internet today.

🧠 **Rancho-Style Analogy**
OSI = "📖 Textbook model" → Great for exams & understanding.
TCP/IP = "⚙️ Working machine" → Real-world internet runs on it.
Think of OSI as **the recipe**, TCP/IP as **the actual food you eat** 🍛.

🌐 **DNS (Domain Name System) — The Contacts App of the Internet**
📌 **What is DNS?**
DNS = Internet's **contacts app** 📱.
Humans remember names (like *bernard*), computers need numbers (like phone numbers = IP addresses).
DNS maps **domain name → IP address**.
Without DNS, you'd have to memorize IPs (like 142.250.72.206 instead of *google.com*).

📌 **How DNS Works (Step-by-Step Journey)**
Stub Resolver (Your Computer)
First checks **cache** (local memory).
If domain already visited, IP is ready.
If not, asks a DNS server.
Recursive DNS Server (like Google 8.8.8.8)
Acts like "a guy who knows a guy."
If it has cached IP → returns it.
If not, asks higher authorities.
Root Servers (The Mafia Bosses 🕴)
13 sets worldwide, managed by organizations like NASA, DOD, ICANN.
Don't give IP addresses directly.
Only tell: "Ask the .com / .org / .net managers."
TLD (Top Level Domain) Servers
Example: .com, .net, .in, .jp.
They say: "For *networkchuck.com*, Cloudflare is the boss."
Authoritative Name Server (The Real Boss 🎩)
Stores the **zone file** (all DNS records for the domain).
Example: *academy.networkchuck.com → 104.18.42.139*.
Sends the final IP address back.
Back to You
Recursive server caches it.
Stub resolver gets IP.
Browser finally connects to the website. 🚀

📌 **Types of DNS Records**
A Record → Domain → IPv4 address.
AAAA Record → Domain → IPv6 address.
NS Record → Which server is authoritative.
MX Record → Mail server (e.g., Gmail handles *@networkchuck.com*).
PTR Record → Reverse lookup (IP → domain).
CNAME Record → Alias (www.example.com → example.com).
TXT Record → Extra info (used for SPF, DKIM, DMARC for email security).

📌 **DNS Security Issues**
Normal DNS uses **UDP port 53** → plain text → can be sniffed/spoofed.
Hackers or ISPs can see what sites you visit.
Attacks: **DNS spoofing, cache poisoning**.

📌 **Securing DNS**
DNS over HTTPS (DoH) → encrypts DNS queries inside HTTPS.
DNS over TLS (DoT) → encrypts queries via TLS.
DNSSEC → ensures authenticity of responses.
Filtering DNS (e.g., Quad9, Cloudflare) → blocks malicious domains.

📌 **Real-World Notes**
ICANN → Governs DNS globally.
Registrars (like GoDaddy, Squarespace) → where you buy domains.
Can **host your own DNS server** using tools like Pi-hole or AdGuard.
Subdomains (like *academy.*networkchuck.com) → point to different resources.

🧠 **Rancho Analogy**
DNS = Your **contacts app**.
Root servers = **mafia bosses** (they know who's in charge but not the numbers).
TLD servers = **middle managers**.
Authoritative servers = **final boss who knows the number**.
Without DNS → Internet = **a phone without contacts** 📞.

IP Addressing — The Internet's Phone Number System
📌 **What is an IP Address?**
Every device (PC, phone, smart fridge… even toilets 😅) needs an **IP address** to talk on a network.
Think of it like a **phone number** for your device. Without it, no communication.
Example: Your laptop talks to YouTube using IPs, not names.

📌 **How to Find Your IP Address**
Windows → Open CMD → ipconfig
Linux → Open Terminal → ifconfig
Mac → Terminal → ifconfig
Phones → Settings → Wi-Fi → Info → See IP.

📌 **The IP Address Family**
When you check your IP, you'll also see:
Subnet Mask (Netmask) → Tells which part of the IP is the **network** (street name) and which part is the **host** (house number).
Default Gateway (Router) → The "post office" 📮 or "Oprah" that delivers traffic outside your network.

📌 **How Devices Get IP Addresses**
Your **router** automatically assigns IPs using **DHCP**.
Every time a device connects, the router says: "Here's your number!"

📌 **Subnet Mask Trick (Hack)**
IPs are written as **four numbers (octets)**: 192.168.1.20
Subnet mask: 255.255.255.0
Wherever you see **255** → that part of the IP never changes (network portion).
Wherever you see **0** → that part can change (host portion).
Example:
Network = 192.168.1 (same for all devices).
Host = 20, 21, 22 … each device gets its own.

📌 **Network Portion vs Host Portion (Rancho Analogy 🏡)**
Network portion = Street name (e.g., "Private Drive").
Host portion = House number (e.g., "4 Private Drive").
Devices on the **same street** talk directly to each other.
Devices on **different streets** need the **router (default gateway)** to deliver messages.

📌 **Reserved IPs in Every Network**
First IP → **Network address** (e.g., 192.168.1.0) → can't assign.
Last IP → **Broadcast address** (e.g., 192.168.1.255) → used to shout messages to everyone.
One IP → Taken by **router/gateway**.
So, in a /24 network (256 total), you usually get **253 usable IPs**.

📌 **Why This Matters**
Every job in IT (networking, hacking, cloud, security) needs IP/subnetting knowledge.
Without understanding IPs, you can't configure firewalls, VPNs, or troubleshoot networks.

✅ So in short:
IP address = phone number 📞
Subnet mask = tells you which part is street vs house
Router = post office (or Oprah 😅)
Reserved IPs = special cases you can't use

🔐 **Encryption in Computer Networks**
1. Symmetric Encryption (Shared Secret Key)
Same **key** used for **encryption + decryption**.
Both sender and receiver must share this key secretly.
Problem → Key distribution is risky. If someone steals it, all data is exposed.
Advantage → Very **fast and efficient**.
Example → AES, DES.

2. Asymmetric Encryption (Public & Private Keys)
Uses **two keys**:
- **Public Key** → shared with everyone.
- **Private Key** → kept secret by the owner.

If data is encrypted with **public key**, only the matching **private key** can decrypt it.
Much more secure for **key exchange**.
Slower than symmetric encryption.
Example → RSA, ECC.

3. Why Use Both? (Hybrid Approach)
Symmetric → Fast, but unsafe for sharing keys.
Asymmetric → Safe for key exchange, but slow.
Solution → Use **asymmetric encryption to share a symmetric key**, then continue communication with fast symmetric encryption.
This is how **SSL/TLS (HTTPS websites)** work.

🎯 **In 1 line**:
Symmetric = one key, fast but risky 🔑
Asymmetric = two keys, safe but slow 🔐
Real world = **combine both** 🚀

🔐 **Hash Functions in Cryptography**
1. What is a Hash Function?
A **procedure** that converts any input (file/text) → **fixed-size string** (called **digest** or **hash value**).
Unique for each input.
Used for:
- Data integrity check 🛡️
- Password security 🔑
- Detecting unauthorized changes 🚨

2. How Hashing Works
Input file (any format: text, doc, image, etc.) → passed through a **hashing algorithm** (MD5, SHA-1, SHA-256).
Output = **digest** (unreadable & fixed size).
Irreversible → can't get back original data from hash.
⚡ Difference from Encryption:
- Encryption = reversible (decrypt).
- Hashing = one-way, irreversible.

3. Key Qualities of a Hash Function
Fixed Size Output → Input may be 1 word or 1GB file, output length is always same.
- Example: MD5 → 128 bits, SHA-256 → 256 bits.

Avalanche Effect → Small input change → totally different hash value.
Irreversible → Original data cannot be guessed from hash.

4. Real-Life Uses

Passwords in Databases

Plain text password is never stored.
Instead → password's hash is stored.
On login → system hashes your entered password → compares with stored hash.
✅ If match → access granted.

Data Integrity

Example: Downloaded file hash = given hash on website.
If both match → file not tampered.
If mismatch → file corrupted/altered.

5. Example

Input: "ABC" → MD5 hash = 902fbdd2... (128 bits).
Input: "ABC." → Hash value is totally different!
Proves **integrity checking**.

🎯 **In one line**:
Hashing = **Digital fingerprint** of data → fixed size, unique, irreversible, and widely used for integrity + password protection.

Alright, so let's understand in this section the working of **SSL and TLS** and related terms like **hashing, encryption, public keys, private keys, and certificates**—how all of these work so that your sensitive data can be securely transferred over the Internet.
For example, this is your computer connected to the Internet, and you want to use **internet banking**. You simply access the banking website, say banking.com. The website will load, and you will provide your username and password to use internet banking.
This data is sensitive, and when it flows over the Internet, there is a chance that hackers could intercept it. Once they have the username and password, they could misuse it. That's why **secure transfer of sensitive data** is crucial.
This is where **SSL (Secure Socket Layer) and TLS (Transport Layer Security)** come in. They are **cryptographic protocols** that allow us to transfer sensitive and important data securely over the Internet.

SSL vs TLS

TLS is the successor of SSL. TLS is the latest version, while SSL was used earlier. Today, TLS is what we actually use, but people still use the term SSL for security purposes. For example, we still say "SSL certificate," but technically, it's TLS that's being used.
For example, when you access a secure website with **HTTPS**, you will see a **lock icon** in the browser. Clicking on it shows that your connection is encrypted—your communication with the website is secure, so any data you transfer will remain safe.

How SSL/TLS Secures Data

SSL/TLS **encrypts data between the client and the server** to ensure:

Confidentiality – data is only accessible to the intended recipient.

Integrity – data isn't modified during transit.

Authentication – verifying the identity of the parties involved.

Example: Exam Paper Transfer

Suppose a question paper from a reputed exam needs to be securely sent from **Point A (printing area)** to **Point B (exam center)**. We need to ensure:

Confidentiality: Only the exam center can access the paper. No one else should see it in between.

Integrity: The paper shouldn't be modified during transit.

Authentication: The authorized person at the exam center must receive it. Verification like checking ID is necessary before handing over the paper.

Methods SSL/TLS Uses

SSL/TLS achieves these three principles using:

Encryption – ensures confidentiality. Converts plaintext into a coded form (ciphertext) so unauthorized users can't read it.

> **Example:** "DEMO" could be encrypted by shifting each letter three places forward: D → G, E → H, etc.

Hashing – ensures integrity. Converts data into a fixed-length string of characters.

> **Example:** Using **MD5 or SHA algorithms**, the exam paper's content is hashed. The receiver hashes it again and verifies that it matches the original hash.

Certificates – ensures authentication. Verifies the identity of websites or servers.

Encryption Types

1. Symmetric Encryption

Same key is used for **encryption and decryption**.

Example: Shift each letter by 3 forward for encryption, shift 3 backward for decryption.

Fast, efficient, suitable for **large amounts of data**.

Challenge: Key distribution over the Internet can be risky.

2. Asymmetric Encryption

Different keys are used for encryption and decryption (**public and private keys**).

Example: Client encrypts with the public key, server decrypts with the private key.

More secure, but slower, suitable for **small, critical data**.

3. Hybrid Encryption

Combines the strengths of symmetric and asymmetric encryption.

Example: Use asymmetric encryption to exchange the **symmetric session key**, then use symmetric encryption for transferring large amounts of data.

Hashing in Detail

Hashing ensures **data integrity**:

Converts data into a **fixed-length string**, e.g., a checksum or digest.

Example: Original message "DEMO" → hashed to 37.

Receiver hashes received data and compares with the original hash.

If hashes match → data is intact; if not → data has been modified.

MAC (Message Authentication Code): Combines message with a secret key, then hashes it. Provides extra security against tampering by attackers.
Common algorithms:
MD5 – produces 128-bit digest
SHA (SHA-1, SHA-256, SHA-512) – produces varying-length digests
HMAC – hash-based MAC

Certificates
Certificates verify the **identity of websites or servers**.
Certificate Authority (CA): Trusted entity that issues digital certificates to website owners.
Certificates contain:

- **Public key**
- **Digital signature** of the CA
- Organization details, domain name, location, etc.

Browsers check the certificate against a list of trusted CAs to ensure validity.
Certificates expire, so they need **renewal**.
Example CAs: GoDaddy, DigiCert, Let's Encrypt.

SSL/TLS Workflow Summary
Client sends HELLO message → supported TLS/SSL versions and algorithms.
Server responds HELLO → chooses TLS version & cipher suite.
Server sends certificate → contains public key & verified identity.
Browser verifies certificate → ensures it's valid & trusted.
Client generates pre-master secret → encrypts it with server's public key → sends to server.
Server decrypts pre-master secret → now both client & server generate a symmetric **session key**.
Handshake completed → secure communication begins.
From now on, **all data transmitted** is encrypted, maintaining **confidentiality, integrity, and authentication**.

1 Symmetric Encryption (AES)
AES is a popular symmetric encryption algorithm.
Step 1: Create a sample message
echo "Hello Rancho-style!" > message.txt
Step 2: Encrypt the message
We'll use **AES-256-CBC** encryption with a password.
openssl enc -aes-256-cbc -salt -in message.txt -out message.enc
-aes-256-cbc → AES encryption in CBC mode
-salt → adds randomness to make encryption stronger
-in → input file
-out → output encrypted file
The command will ask you for a password. Remember it, because you'll need it to decrypt.

Step 3: Decrypt the message
openssl enc -aes-256-cbc -d -in message.enc -out message_decrypted.txt
-d → decryption
-in → encrypted file
-out → decrypted output file
Check the decrypted message:
cat message_decrypted.txt
You should see:
Hello Rancho-style!

2️⃣ Asymmetric Encryption (RSA)

Asymmetric encryption uses **public/private key pairs**.
Step 1: Generate RSA keys
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048
openssl rsa -pubout -in private_key.pem -out public_key.pem
private_key.pem → keep secret
public_key.pem → shareable
Step 2: Encrypt a message using the public key
openssl rsautl -encrypt -inkey public_key.pem -pubin -in message.txt -out message_rsa.enc
-pubin → input is a public key
Step 3: Decrypt the message using the private key
openssl rsautl -decrypt -inkey private_key.pem -in message_rsa.enc -out message_rsa_decrypted.txt
Check the decrypted message:
cat message_rsa_decrypted.txt
You should see your original message:
Hello Rancho-style!

✅ Key Takeaways

Symmetric encryption is faster and suitable for large messages, but requires safe key sharing.
Asymmetric encryption solves the key-sharing problem but is slower and usually used for small messages or encrypting session keys.
OpenSSL is a powerful tool to practice real-world encryption/decryption.

Wireshark Masterclass – Lesson 1: Setup & Customization
1. Profiles – Your Personal Wireshark Setup
What is a profile?
A profile is like your personal car seat settings—columns, filters, colors, and dissectors saved for specific tasks.
Why use it?
Different tasks (TCP troubleshooting, VoIP, TLS) need different layouts.
How to create one:

Go to the lower-right corner → right-click → New Profile.
Name it (e.g., Wireshark Masterclass).
All your settings will now save here.

2. Adjusting Screen Layout

Make text readable: Use the magnifying glass to zoom in.

Fix column overlap: Drag column edges to adjust width.

Packet layout preference:

Default: Packet bytes below packet details.

Personal: Packet bytes to the right (better for header analysis).

How:

Mac: Wireshark → Preferences → Layout → select preferred pane arrangement.

Windows: Edit → Preferences → Layout.

3. Visualizing Packet Headers

Packet diagram view: Shows Ethernet, IP, TCP headers visually.

Right-click → Show Field Values: Extracts values from packets for easier reading.

Switch back to normal packet bytes: Preferences → Layout → Pane 3 → Packet Bytes.

4. Adding Columns

Delta time column: Shows time between packets—helpful for troubleshooting.

Preferences → Columns → + → Name: Delta → Type: Delta time displayed → OK → Drag next to Time column.

Other columns:

Example: TCP segment length → Right-click field → Apply as column.

Remove default Length column if needed.

5. Coloring Rules

Purpose: Make important packets stand out (e.g., TCP SYN, errors).

Example: Color TCP SYN green:

View → Coloring Rules → + → Name: TCP SYN.

Filter: tcp.flags.syn == 1.

Background color: bright green → OK.

Place rule below "Bad TCP" to keep retransmissions red.

Tip: Color rules = your personal troubleshooting style. "Your way or the highway!"

6. Display Filter Buttons

Purpose: Quickly apply filters without typing syntax every time.

How:

Right-click a packet → Prepare as filter → Selected.

Add as button: Click +, label it (e.g., TCP SYN), paste the filter → OK.

Click button anytime to apply filter instantly.

7. Summary of Lesson 1
Adjust **screen layout** for comfort.
Switch between **packet bytes** and **header views**.
Add **columns** (Delta time, TCP segment length).
Apply **coloring rules** for key packets.
Create **display filter buttons** for quick access.
Save everything in a **profile** for different troubleshooting scenarios.

💡 **Pro Tip:** Start small. Focus on one profile with basic filters, columns, and colors. As you gain confidence, add advanced filters and custom layouts.

1. What is Nmap?
Nmap (Network Mapper) is a free tool used for scanning and enumerating networks.
Hackers and IT pros use it to discover:

- Live hosts
- Open ports
- Running services
- Vulnerabilities
- Operating systems

2. Basic Nmap Commands
Ping scan: Check which hosts are alive.
nmap -sP 10.7.1.0/24
TCP scan: Check for open ports using a full three-way handshake.
sudo nmap -sT -p 80,443 10.7.1.0/24
Stealth scan (half-open SYN scan): Avoid detection by firewalls.
sudo nmap -sS -p 80,443 10.7.1.0/24

3. How Nmap Works
Uses **TCP/IP protocol** to communicate.
Three-way handshake:

- SYN: Client asks to connect
- SYN-ACK: Server acknowledges
- ACK: Client confirms

Stealth scan skips completing the handshake to avoid detection.

4. Advanced Scanning Features

OS detection:
sudo nmap -O <target>
Aggressive scan: Combines OS detection, version detection, script scanning, and traceroute.
sudo nmap -A <target>
Decoys: Send scan traffic from multiple IPs to hide your real IP.
sudo nmap -D <decoy1,decoy2,...> <target>
Nmap Scripting Engine (NSE): Automates vulnerability checks and advanced info gathering.
sudo nmap --script vuln <target>

5. Real-World Uses

Scanning your own network to find live devices.
Checking open ports for web servers or services.
Detecting OS and service versions for penetration testing.
Automating vulnerability checks with scripts.
Learning cybersecurity fundamentals in a lab environment (like VulnHub VMs).

6. Best Practices

Use a lab environment (like Kali Linux + VulnHub) for practice.
Start with small scans before running aggressive scans.
Be aware of legal and ethical boundaries—scanning networks you don't own can be illegal.

1. Install FoxyProxy

Open Firefox → search for **"FoxyProxy Standard"** → click **Add to Firefox**.
Allow it to run in private windows.

2. Configure Burp Suite

Open **Burp Suite** (Temporary Project → Use Burp defaults → Start Burp).
Go to **Proxy → Proxy Settings → Proxy Listeners**.
Note the **IP (127.0.0.1)** and **Port (8080)**.

3. Configure FoxyProxy

Click the FoxyProxy extension → Options → Proxies → Add.
Enter the **IP** and **Port** from Burp Suite.
Save and enable FoxyProxy.

4. Import Burp's CA Certificate

In Firefox, visit http://burp → download the **CA certificate**.
Open Firefox Settings → Privacy & Security → Certificates → View Certificates → Authorities → Import.
Select the certificate → check "Identify websites" and "Identify email users" → OK.

5. Intercept Traffic
In Burp Suite → turn **Intercept On**.
Enable FoxyProxy in Firefox → browse any website.
Traffic now appears in Burp Suite for interception.
Tip:
To intercept local sites (e.g., http://localhost), go to about:config → search network.proxy.allow_hijacking_localhost → set to **true**.

✅ **Key Takeaways:**
FoxyProxy routes browser traffic to Burp Suite.
Burp's CA certificate is necessary to intercept HTTPS traffic.
Always intercept your **own traffic**; intercepting public websites is illegal.

Netcat (NC) Overview
What it is: A versatile networking utility in Linux/Windows to read/write network connections over **TCP/UDP**.
Common Uses:
- Banner grabbing / OSINT
- File transfers ("Dead Drops")
- Chat server/client setup
- Remote shells / remote code execution
- Port scanning

Command Basics:
- nc = netcat
- -v → verbose (see what's happening)
- -w → wait time (timeout in seconds)
- -l → listen mode
- -p → specify port
- -e → execute a program (like bash shell)

1. Banner Grabbing (OSINT)
Goal: Get info about a server/web service (e.g., server type).
Example:
nc -v <Target_IP> 80
You can type HEAD / HTTP/1.0 to get server info.
Helps admins see what info their web servers expose.

2. Chat Server / Client
Server (Kali Linux):
nc -l -p 5555 -v
Client (Ubuntu):
nc <Server_IP> 5555 -v
Anything typed in client appears on server.
Useful for internal communication testing.

3. File Transfer ("Dead Drop")
Server (Kali Linux):
nc -l -p 12241 -w 60 -v < netcat.txt
Client (Ubuntu):
nc <Server_IP> 12241 -w 2 -v > netcat.txt
Server "drops" a file; client picks it up later.
Can **append** instead of overwrite using >>.

4. Remote Shell / Remote Code Execution
Server (Kali Linux):
nc -l -p 12241 -e /bin/bash -v
Client (Ubuntu):
nc <Server_IP> 12241 -v
Client now has **full shell access** to server.
⚠️ Highly sensitive! Only do this in **lab/authorized environments**.

5. Port Scanning
Quick way to check which ports are open on a host:
nc -v -z <Target_IP> 1-13000
-z → zero-I/O mode (just scanning).
Shows open ports (e.g., your Netcat listener port 12241).

Tips / Housekeeping
Use nc -h or man nc for full command options.
Always verify IP addresses (ifconfig) and file locations (pwd, ls).
Combine commands with && to check multiple things at once:
whoami && hostname && pwd && date

✅ **Key Takeaways**
Netcat is like the Swiss Army knife for networking.
Can be used for **testing, troubleshooting, or learning pentesting**.
Most commands are simple but powerful; **always test in controlled environments**.